IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 12, 2020, accepted May 20, 2020, date of publication May 25, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997387

Insights Into Efficient k-Nearest Neighbor Classification With Convolutional Neural Codes

ANTONIO-JAVIER GALLEGO, JORGE CALVO-ZARAGOZA, AND JUAN RAMÓN RICO-JUAN
Pattern Recognition and Artificial Intelligence Group, Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain

Corresponding author: Juan Ramón Rico-Juan (juanramonrico@ua.es)

This work was supported in part by the Spanish Ministerio de Economía, Industria y Competitividad through Project HISPAMUS under Grant TIN2017-86576-Research, and in part by the EU FEDER funds.

ABSTRACT The increasing consideration of Convolutional Neural Networks (CNN) has not prevented the use of the k-Nearest Neighbor (kNN) method. In fact, a hybrid CNN-kNN approach is an interesting option in which the network specializes in feature extraction through its activations (Neural Codes), while the kNN has the advantage of performing a retrieval by means of similarity. However, this hybrid approach also has the disadvantages of the kNN search, and especially its high computational cost which is, in principle, undesirable for large-scale data. In this paper, we present the first comprehensive study of efficient kNN search algorithms using this hybrid CNN-kNN approach. This has been done by considering up to 16 different algorithms, each of which is evaluated with a different parametrization, in 7 datasets of heterogeneous composition. Our results show that no single algorithm is capable of covering all aspects, but rather that each family of algorithms is better suited to specific aspects of the problem. This signifies that Fast Similarity Search algorithms maintain their performance, but do not reduce the cost as much as the Data Reduction family does. In turn, the Approximated Similarity Search family is postulated as a good option when attempting to balance accuracy and efficiency. The experiments also suggest that considering statistical transformation algorithms such as Linear Discriminant Analysis might be useful in certain cases.

INDEX TERMS Classification, convolutional neural networks, efficient searches, nearest neighbor searches, neural codes.

I. INTRODUCTION

The k-Nearest Neighbor (kNN) classifier is one of the classical schemes for supervised learning tasks [1] and it is still considered in current research, as discussed in a recent retrospective by Kuncheva [2]. Most of its popularity originates from its conceptual simplicity and straightforward implementation, which are well suited to many disparate duties. This algorithm hypothesizes about the category of a given input in the feature space by following a defined similarity measure to query its k-nearest neighbors in the training set and applying a plurality vote scheme to select the most common category.

The performance of the classifier, therefore, improves as the training set increases, since it has been demonstrated that its error is bounded by twice the Bayes error when the number of training samples approaches infinity [3]. Since the beginnings of information-related technologies, data production has been reported to be constantly growing [4], and this effect has become more remarkable in recent years. A kNN classifier may, therefore, be able to exploit these large-scale sources of information in order to improve classification performance.

As a representative example of instance-based algorithms, the kNN classifier does not carry out an explicit generalization process (i.e., building a model) on the initial training data but directly considers those samples for classification [5]. This behavior is especially useful when you need, in addition to knowing the category of a new sample, a list of other similar samples that belong to the historical (training set). This might be helpful in some contexts so that human experts focus their attention on these samples to make decisions or detect outliers. When other classification algorithms are used, this inspection would be much more complex or impossible. Some recent works in areas such as health [6] use kNN to

The associate editor coordinating the review of this manuscript and approving it for publication was Orazio Gambino.

improve the accuracy results with respect to SVM (Support Vector Machines) or ANN (Artificial Neural Networks). Moreover, in a real application, it would be interesting for specialists to know which patient records are the most similar in order to analyze them in detail.

Recent advances in feature learning, namely Convolutional Neural Networks (CNN), have made a breakthrough as regards the ability to learn suitable features for classification. That is, rather than resorting to heuristic processes for feature extraction, these networks are trained to infer a suitable representation for the task in hand from the raw input signal. Some authors have, however, shown that it is also interesting to use CNN only as feature extractor engines, i.e., feeding the network with raw data and taking one of the intermediate representations, most typically the second-to-last layer output, as features for the classification task [7], [8].

The kNN method may obtain more complex decision boundaries than the last layer of a CNN, which usually estimates a normalized linear function. However, it requires the features and/or the distance considered to be adequate for the task. Taking into account that CNN and kNN are totally complementary in terms of feature extraction and decision boundaries, it is interesting to consider the hybrid approach in which both strategies can exploit their potential and mitigate each others' drawbacks. Forwarding a raw input through the network makes it possible to obtain an appropriate representation from the last layers. This representation is a numerical vector that is also referred to as Neural Codes (NC) [9].

Nevertheless, since the kNN classifier needs to compute a distance between the input sample and every single sample in the training data, this entails low efficiency as regards both classification time and memory usage. This is the main drawback of this approach, which becomes an insurmountable obstacle when considering large-scale training corpora.

All the strategies with which to alleviate these issues that have been proposed to date have been evaluated in a conventional scenario as regards the kNN. In addition, to the best of our knowledge, no work compares all the families of strategies whose objective is to improve the k-nearest neighbor search. Many of these strategies normally seek to improve efficiency, which is often achieved at the cost of worsening classification accuracy. Since the combined use with CNN modifies this paradigm, a comprehensive study has been carried out in order to provide some insights into the use of efficient strategies for kNN hybridized with NC. We trust that this study will enable the clarification of which the best option is, and under what circumstances, in order to make the use of the CNN-kNN hybrid approach feasible with large amounts of data.

In summary, the paper makes the following contributions:

1) A detailed formalization of the combination between kNN and NC.
2) A comprehensive study of how to perform such combination as regards efficiency and accuracy.
3) Thorough experiments with different types of CNN configurations, NC sizes, efficient kNN search approaches, heterogeneous datasets, and a wide set of parameters.
4) Interpretation of the experimental results that goes beyond listing performance values, which we expect to be useful for researchers working in this field.

Our paper first reviews the background to the field in Section II. In Section III, we then go on to explain the methodology behind the classification with a hybrid CNN-kNN approach. The experimentation setup followed to evaluate the several options considered in a number of datasets is described in Section IV. The results obtained, along with a thorough analysis of them, are presented in Section V. Finally, we summarize the main conclusions in Section VII, in addition to providing some ideas for future work.

II. BACKGROUND

As a representative example of instance-based classification, the kNN classification rule is generally highly inefficient: since no model is built from the training data, every training sample is consulted at the time of classifying an input query. This condition has two clear implications: on the one hand, a considerable amount of storage requirements, and on the other, a high computational cost. Some variants of the kNN include a training process, such as the work of Zhang *et al.* [10], in which a model is built in order to infer the optimal k for each sample. However, this does not reduce the cost when predicting a sample.

These shortcomings have been widely analyzed in literature and several strategies with which to tackle them have been proposed. In general, they can be divided into three categories: Fast Similarity Search (FSS) [11], Approximated Similarity Search (ASS) [12], and Data Reduction (DR) [13].

FSS is a family of methods whose performance is based on the creation of search models for fast prototype retrieval in the training set. These strategies are generally further subdivided into indexing algorithms [14] and the Approximating and Eliminating Search Algorithm (AESA) family [15]. The former family represents the set of algorithms that iteratively partition the search space and build tree structures for an efficient search; for a new element to be classified, a search of the tree takes place in order to select the proper space partition (leaf node in the tree) to subsequently perform an exhaustive search of the prototypes in that region; this implies that only one subset of the total number of examples has to be queried to classify a new instance. Some examples of these methods and structures are *KD Trees* [14], *Ball Trees* [16], and *Metric Trees* [17], amongst others. The problem, however, is that they are extremely sensitive to the curse of dimensionality, and they additionally require that input data be represented as feature vectors. Note that, in the case of our hybrid CNN-kNN approach, these drawbacks are mitigated because Neural Codes are in fact numerical feature vectors, and their dimension can be adjusted when configuring the network. AESA algorithms, on the other hand,

demonstrate their potential with structured data (such as strings, trees, or graphs) because they require only a metric space, *i.e.* that in which a pairwise distance can be defined. These strategies make use of pre-computed distances and the triangle inequality to discard prototypes. The main disadvantage of these algorithms is that they typically deal with searches involving $k = 1$ and become memory-inefficient with large-scale data. In addition to these techniques, there are also studies that consider specific computing engines like Apache Spark for the highly efficient performance of similarity searches [18], [19].

ASS approaches work on the premise of searching for sufficiently similar prototypes to a given query in the training set, rather than retrieving the exact nearest instance. This improves the efficiency of the algorithm at the cost of decreasing the classification accuracy. When large datasets are present, the ASS framework emerges as a suitable option to consider because the possible drawbacks, such as the accuracy loss produced by not retrieving the actual nearest prototype, are mitigated by the huge amount of information available. Some particularly successful principles within this family are the use of hashing techniques to codify the prototypes of the training set. Typical examples comprise the Local Sensitive Hashing (LSH) forest [20], which is based on different distances and demonstrable improvement of the search scheme, Spectral Hashing [21], which was the first research that considered consistency with Hamming codes to find a function whose similar elements are mapped to similar hash codes with a small number of bits, or Product Quantization [22], which divides the features space into disjoint sub-spaces represented by vectors that are clustered separately, each of which can be coded with logarithmic complexity. A different approach is the use of data clustering in order to restrict the search to a specific portion of the space [23], [24]. Approximate KD Trees have been also considered within this family of techniques (e.g., the Fast Library for Approximate Nearest Neighbors [25]).

DR comprises a set of strategies whose objective is to reduce the size of the initial training set while maintaining the same recognition performance [13]. The two most common approaches are Prototype Generation and Prototype Selection [26]. The former creates new artificial data to replace the initial set more efficiently, while the latter simply selects certain elements from that set that are sufficiently representative. The *Condensed Nearest Neighbor* [27] was one of the first techniques developed for this purpose, yet several proposals can be found in literature in both the selection [28] and generation [29] paradigms. More recently, there have been a number of new proposals, such as Instance Reduction Algorithm using Hyperrectangle Clustering [30] that reduces non-border instances using a hyper-rectangle technique with the min-max points obtained by a clustering algorithm, Reduction through Homogeneous Clusters [31], which is based on a fast cluster pre-processing procedure that creates homogeneous clusters to select their centroids as representative samples, or Edited Natural Neighbor [32], which aims to eliminate noise patterns based on the concept of natural neighbor, with no parameters and whose selection is made in an adaptive way. In any case, the main problem with these methods is that they generally imply a significant loss of accuracy in the classification [13]. Various strategies with which to resolve these deficiencies have, therefore, been proposed, such as considering boosting schemes [33], merging feature and prototype selection by means of genetic algorithms [29], [34], or considering the results of these reduction algorithms as only a means of constraining the categories to be taken into account by the conventional kNN [35].

III. METHODOLOGY

Convolutional Neural Networks (CNN) are multi-layer architectures designed to extract high-level representations of a given input. They have dramatically improved the state of the art as regards image, video, speech, and audio recognition tasks [36]. When trained for supervised classification, the layers of a CNN are eventually able to extract a set of features that are suitable for the task at hand. These features are obtained by forwarding a raw input through the network, in which the last layer merely learns a linear mapping to provide each possible category of the classification domain with a probability.

Its high generalization power allows transfer learning to be used to apply CNN models trained on a domain to a different task in which the data are similar but the categories are different [37]. This transfer can be done by fine-tuning the pre-trained network in the new dataset [38]. Alternatively, it can also be performed by using the CNN as a feature extractor, by forwarding samples through the network to obtain the activations from one of the last hidden layers, which is usually a fully-connected or a pooling layer. These representations are also referred to as Neural Codes (NC) [9].

Extracting neural codes and then applying Support Vector Machines (SVM) or kNN is a common transfer learning technique [8], [39]. However, a lazy classification method such as kNN is preferred to SVM for similarity search tasks, in which the interest lies not only in the category but also in obtaining similar prototypes. At the inference stage, NCs from the last hidden layer (the layer before the output or classification layer) are extracted to perform a kNN search on the training set. Nevertheless, the power of this hybrid CNN-kNN approach is initially mitigated by the aforementioned drawbacks of the kNN classifier itself.

In this paper, we analyze the use of efficient kNN strategies in the context of the hybrid approach introduced above. The proposed methodology is illustrated in Fig. 1. The first step is to train a CNN in a supervised fashion by providing pairs containing the input samples and their labels. Let $T = \{(x_1, y_1), (x_1, y_1), \ldots, (x_m, y_m)\}$ be the set of M training samples where each sample x_i has an associated label y_i from the set of possible categorical labels $Y = \{1, \ldots, L\}$. The CNN (denoted by G) implements a function $G : X \rightarrow Y$ that classifies an instance $x \in X \subset \mathbb{R}^D$ into a label of Y. The process of training G consists of adjusting the set of network weights

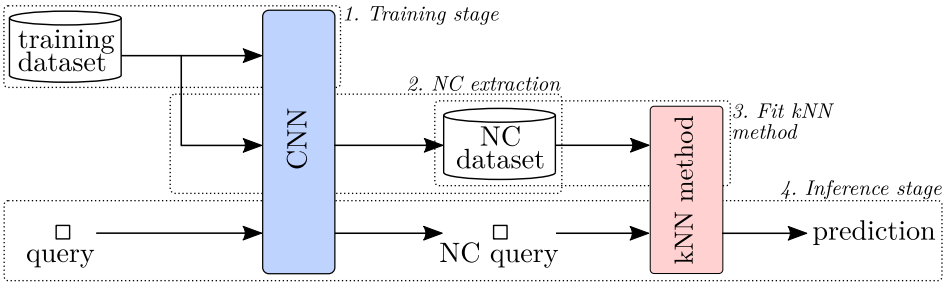


FIGURE 1. Graphical scheme of the classification methodology considered in the present work.

using the training set T to minimize the classification error according to a given loss function $\mathcal{L}$ [40] and considering any conventional means for network optimization such as stochastic gradient descent [41].

Once G has been trained, it is used to obtain the set of encoded training data. This is done by forwarding the samples of T through the network to extract the feature vectors from a user-defined feature layer (denoted by G^F). That is, $G^F : X \rightarrow \mathbb{R}^N$ maps an instance x from a D-dimensional representation onto a new N-dimensional feature space $X^F \in \mathbb{R}^N$ representation. This new representation, also referred to as Neural Codes (NC) [9], is stored in the set T_{NC} and used to build the efficient kNN search strategy to be evaluated.

In general, a kNN search hypothesizes about the category of a given input or query q by following a defined similarity measure to query its k-nearest neighbors in the training set T. The query q is classified by a plurality vote within its neighbors, with the query being assigned to the most common class among its k nearest neighbors. Therefore, kNN can be defined as:

$$\text{mode}\left(Y_k\left(\underset{x_i \in T}{\arg\min_k}\{d(q, x_i)\}\right)\right)$$

where $d(q, x_i)$ denotes the distance between the prototypes q and x_i. Note that the distance considered is the Euclidean one because, as described above, the features compared (in this case, the NC) are numerical feature representations. Other distances could also be considered, like Manhattan or Mahalanobis [42], or even other types of structural-based measures, such as SimRank [43], C-Rank [44] or HeteRank [45]. However, the Euclidean distance has been used in all cases for two reasons. On the one hand, some of the efficient search methods compared exploit properties of distance functions (such as the triangle inequality), for this reason, *pseudometrics* as the referenced structural-based measures cannot be applied. On the other hand, since the implementation of some of these methods is based on the Euclidean distance, we decided to keep the same metric for all cases to make a fair comparison.

As can be seen from the equation above, kNN performs an exhaustive search for q in the whole training set T. This implies a poor search efficiency that worsens as the size of T increases. The kNN-based efficient search methods simply try to reduce the number of distances calculated by using some of the strategies previously described in Section II.

To summarize, Algorithm 1 shows the formalization of the training process using pseudocode. The algorithm receives as input the training set T, the CNN topology G, the efficient kNN search method S, and the training parameters (epochs and batch size), and returns the trained network as output along with the new training set T_{NC}, and the search method S prepared to search within this set. It is important to note that this algorithm uses only the training set T (i.e., it does not need the test set). Therefore, it can be performed as a pre-process before the inference stage without affecting the efficiency of the search, but rather the opposite.

Algorithm 1 Training Stage

Input : $T \leftarrow \{(x_i, y_i)\}_{i=1}^{|T|}$
$G \leftarrow$ CNN topology
$S \leftarrow$ Efficient kNN search method
$e, b \leftarrow$ Epochs, batch size
Output: G, S, T_{NC}
1 $G \leftarrow$ Fit G with $\{T, e, b\}$
2 $T_{NC} \leftarrow G^F(T)$
3 $S \leftarrow$ Fit S with T_{NC}

The classification of new samples comprises a series of steps that make use of the data prepared during Algorithm 1. Specifically, Algorithm 2 shows the formalization of the classification process using pseudocode. The test sample q is forwarded through the pre-trained network G to transform its original features by using the feature layer G^F to obtain its NC (stored in q_{NC}). The kNN strategy S that has been built during the training stage is then used to perform the classification.

Algorithm 2 Inference Stage

Input : q, G, S, T_{NC}
Output: N_q
1 $q_{\text{NC}} \leftarrow G^F(q)$
2 $N_q \leftarrow S(q_{\text{NC}}, T_{\text{NC}})$

In this work we wish to provide a detailed analysis of the advantages and disadvantages of each of the existing options as regards performing an efficient kNN classification

with NC. In this context, it is also interesting to analyze how the different configurations and parameters affect both the accuracy (which is strongly related to the accuracy of the underlying CNN) and the efficiency of the approach. For instance, in the case of the dimension of the NC, the accuracy of the network may improve for a particular size, which may not be optimal for the efficiency of the subsequent kNN search.

In the proposed scheme, it is possible to adjust the dimension of the NC by changing the size of the feature layer G^F from which the NCs are extracted. As previously stated, the last hidden layer of the CNN is usually used as the feature layer [9], [46], which typically consists of a fully connected layer with N artificial neurons. Therefore, varying the dimension of the NCs simply consists of adding or removing neurons from this layer and retraining the network to adjust the weights to the new dimension.

Furthermore, it is necessary to determine the size of the feature layer. In many cases, this is not empirically evaluated and the size provided by the network configuration is directly used, so high-dimensional NCs are often used (e.g., 4096 [9]). However, it is to be expected that this size will directly affect both the classification performance and the efficiency of the search [47]. For example, when using FSS techniques such as KDTree [14], if the dimensionality of the data is very large, it can degrade the number of searches performed by the method, thus making it as inefficient as the performance of an exhaustive search [48].

The dimension of the NC and many other issues will be addressed after conducting comprehensive experiments considering different network configurations, datasets, and classification scenarios. The experimental plan used to carry out this analysis is described in the following section.

IV. EXPERIMENTAL SETUP

A. DATASETS

The configuration presented was evaluated with different datasets selected to depict different numbers of features and samples. Our evaluation specifically comprises the following seven datasets of images (summarized in Table 1):

- *United States Postal Office* (USPS) [49] and *MNIST* [50] are datasets of binary images depicting handwritten digits. Each comprises 10 classes (from 0 to 9).
- *Handwritten Online Musical Symbol* (HOMUS) [51] depicts binary images of isolated handwritten music symbols collected from 100 different musicians.
- *NIST SPECIAL DATABASE 19* (NIST) of the National Institute of Standards and Technology [52] consists of a dataset of isolated characters.
- CIFAR-10 and CIFAR-100 [53] are standard object recognition datasets for the computer vision community. They consist of 32×32 color images extracted from the *80 million tiny images* dataset [54] and containing 10 and 100 different categories, respectively.
- MIRBOT is a collaborative application for object recognition using a mobile device [55]. The data collected consists of color images of varying sizes, which are rescaled here to 224×224. The application establishes a hierarchy of classes, depending on the level of detail, with a varying number of samples for each one. In this work, we consider the 100 classes with more that are most representative (MIRBOT-100).

TABLE 1. Description of the image datasets used in the experimentation, including the number of instances, the number of classes, and the input shape of the samples. The input shape is denoted by $c \times h \times w$, where c is the number of channels of the image (1 for binary or grayscale images and 3 for color images), h is the height and w is the width.

Name	No. of instances	Classes	Input shape
USPS	9 298	10	$1 \times 16 \times 16$
MNIST	70 000	10	$1 \times 28 \times 28$
HOMUS	15 200	32	$1 \times 40 \times 40$
NIST	44 951	26	$1 \times 32 \times 32$
CIFAR10	60 000	10	$3 \times 32 \times 32$
CIFAR100	60 000	100	$3 \times 32 \times 32$
MIRBOT100	20 370	100	$3 \times 224 \times 224$

Concerning the pre-processing of the input data, the values of the pixels from MNIST, HOMUS, and NIST images are divided by 255 for normalization, whereas the mean image is subtracted from the CIFAR (10 and 100) and MIRBOT images. USPS data are already normalized at their origin.

B. CONVOLUTIONAL NEURAL NETWORK MODELS

The hybrid classification scheme requires the definition of a CNN network configuration. Outperforming the state of the art as regards the previous datasets is not, however, within the scope of this paper. We shall, therefore, consider network models that have been proven to deal well with the corpora in order to properly evaluate the effectiveness and the efficiency of the hybrid CNN-kNN scheme. The details of the CNNs for each dataset are provided in Table 2.

In all cases, the last hidden layer of all the networks consists of a fully-connected layer with N neurons, from which the NC will be extracted. In the experiments, the implications of the parameter N will be evaluated empirically. At the time of training, all these configurations are obviously added with a *Softmax* layer of L neurons — where L is the number of labels or categories in the dataset — from which the classification is obtained.

The comparison of the hybrid approach CNN-kNN against the individual kNN or CNN classifiers has already been addressed in some previous works [59]–[61]. Therefore, we shall skip this question to focus our experiments on the efficient k-nearest neighbor search under the hybrid paradigm, which is the most novel aspect of the present work.

C. EFFICIENT kNN STRATEGIES

Given that there are many different approaches for an efficient kNN search, we have selected a set of representative strategies from the different families of algorithms that were introduced in Section II. For the sake of comparison, the conventional kNN search (brute force) has also been included in the experiments.

TABLE 2. CNN network configurations considered in order to obtain NC representations from each dataset. Notation ***Conv***(***f***, ***w***, ***h***) stands for a layer with ***f*** convolution operators of size ***w*** × ***h*** pixels, ***FC***(***n***) represents a fully-connected layer of ***n*** neurons, ***Drop***(***d***) implements a dropout stage with a value of ***d***% and ***MaxPool***(***w***, ***h***) stands for the max-pooling operator of dimensions ***w*** × ***h*** pixels.

Dataset	CNN configuration						
USPS MNIST	Conv(32,3,3)	Conv(32,3,3) MaxPool(2,2) Drop(0.25)	FC(N) Drop(0.5)				
HOMUS NIST	Conv(256,3,3) MaxPool(2,2) Drop(0.2)	Conv(128,3,3) MaxPool(2x2) Drop(0.2)	Conv(128,3,3) Drop(0.2)	Conv(64x3x3) Drop(0.2)	FC(512) Drop(0.1)	FC(256) Drop(0.1)	FC(N) Drop(0.1)
CIFAR10	VGG16 [56]		FC(N)				
CIFAR100	ResNet50 [57]		FC(N)				
MIRBOT100	Xception [58]		FC(N)				

- **Fast Similarity Search (FSS)**: In terms of this family of space partitioning techniques, we have selected the KDTree [14] and BallTree [16] methods, which have been configured to evaluate 10, 20, 30, and 40 prototypes in each leaf node.
- **Approximate Similarity Search (ASS)**: With regard to approximate search approaches, we have principally considered the use of hashing techniques, such as Local Sensitive Hashing (LSH) [20], Spectral Hashing (SH) [21] and Product Quantization (PQ) [22]. We also include the Clustering-based k-Nearest Neighbor (ckNN) algorithm [24] since it is representative of the use of clustering methods for the purpose in hand. In this case, we evaluate the algorithm with its proposed automatic selection of the number of clusters, in addition to fixed values of 25, 50, 75, 100, 200, and 500.
- **Data Reduction (DR)**: We assessed two different options for this particular family of approaches: on the one hand, we considered the Reduction through Homogeneous Clusters (RHC) algorithm [31], while on the other, we tested the meta-algorithm kNNc [35]. This algorithm receives a DR method as a parameter and considers its reduced set to restrict the search to the c-nearest classes of the query. We evaluated the parameter c for the values 1 (which is equivalent to performing the classification with only the base DR algorithm), 2, and 3. The set of base DR algorithms considered is listed below:
  - Classical algorithms: Condensing Nearest Neighbor [27], Editing Condensing Nearest Neighbor [62], and Fast Condensing Nearest Neighbor [63].
  - Rank methods: Farther Neighbor and Nearest to Enemy [64], and Instance Rank based on Borders [65].
  - Heuristic methods: Decremental Reduction Optimization Procedure 3 [66] and Iterative Case Filtering Algorithm [67].

A summary of the strategies considered is provided in Table 3. The interested reader is referred to the referenced articles for further details on the operation of the strategies and the meaning of their parameters.

Furthermore, all the methods have been tested with different values of the parameter k of the kNN search, specifically $k = 1, 3, 5,$ and 7.

D. EVALUATION

In order to analyze the impact of the different strategies for kNN classification, we take into account both their accuracy and their efficiency.

Given that some of the datasets are not evenly balanced, the accuracy metric used for evaluation is the weighted average of the F-measure (Fm) scores of each class. Fm is a widely used metric in information retrieval and class imbalance problems [68]. Taking one class as *positive* and the rest as *negative*, at a time, the Fm can be defined by means of precision and recall as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Fm} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where TP denotes the number of true positives, FP denotes the number of false positives, and FN denotes the number of false negatives. Given that the ground-truth of our data is the true category of each sample, we can easily compute these metrics comparing the ground-truth category with the category determined by the kNN strategy.

In terms of efficiency, the execution time would be an interesting measure to consider. The problem is that it contains a high level of subjectivity and imprecision that depends on many factors that are not related to the goodness of the strategies considered: implementation, programming language, underlying computing architecture, and so on. In order to measure the efficiency more objectively, we shall consider the algorithmic cost of each strategy as $O(DN)$, where D is the number of distances to be computed and N is the dimension of the samples. Note that the distance considered is always the Euclidean distance, because NC is a numerical feature representation. The proposed cost is, therefore,

TABLE 3. Summary of the different efficient *k*NN strategies considered in this work, along with their identifiers, algorithmic family and set of parameters. In all cases, the methods have been tested with different values of the parameter *k* of the *k*NN search, specifically $k = 1, 3, 5,$ and 7.

Algorithm	Identifier	Family	Parameters evaluated
k-Nearest Neighbor	*k*NN	-	
KDTree [14]	KDTree	FSS	leaf $\in \{10, 20, 30, 40\}$
BallTree [16]	BallTree	FSS	leaf $\in \{10, 20, 30, 40\}$
Local Sensitive Hashing [20]	LSH	ASS	trees $\in \{10, 20, 30, 40\}$
Spectral Hashing [21]	SH	ASS	nbits $\in \{40, 80, 100, 120\}$
Product Quantization [22]	PQ	ASS	nsubq $\in \{1, 2, 4, 8\}$
Clustering-based k-Nearest Neighbor [24]	ckNN	ASS	c $\in \{$auto, 25, 50, 75, 100, 200, 500$\}$
Reduction through Homogeneous Clusters [31]	RHC	DR	
Condensing Nearest Neighbor [27], [35]	CoNN	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$
Editing Condensing Nearest Neighbor [35], [62]	ECNN	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$
Fast Condensing Nearest Neighbor [35], [63]	FCNN	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$
Farther Neighbor [35], [64]	FN	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$, r $= 0.1$
Nearest to Enemy [35], [64]	EN	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$, r $= 0.1$
Instance Rank based on Borders [35], [65]	IRB	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$, r $= 0.1$
Decremental Reduction Optimization Procedure 3 [35], [66]	DROP3	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$
Iterative Case Filtering [35], [67]	ICF	DR	k $\in \{1, 3, 5, 7\}$, c $\in \{1, 2, 3\}$

closely related to the actual computational cost of performing a classification with these strategies. In our results, the actual cost reported will be that normalized by the highest cost (that obtained by performing all possible distances with the largest NC size), signifying that the value remains in a range between 0 and 100 (%).

These measures (Fm and cost) allow us to analyze the performance of each of the strategies considered. Nevertheless, no comparison between the whole set of alternatives can be established in order to enable us to determine which is the best. The problem is that these strategies attempt to minimize the computational cost at the same time as they attempt to increase accuracy. These two goals are, quite often, contradictory, and improving one of them consequently implies a deterioration in the other. From this point of view, efficient *k*NN classification can be seen as a Multi-objective Optimization Problem (MOP) in which two functions are optimized at the same time: accuracy and efficiency. The common means employed to evaluate this kind of problems is the *non-dominance* concept. One solution is said to dominate another if, and only if, it is better or equal in each goal function and, at least, strictly better in one of them. The best solutions (there might be more than one) are consequently those that are non-dominated.

The strategies considered will, therefore, be evaluated by assuming a MOP scenario in which each of them is a 2-dimensional solution defined as (Fm,cost). In order to analyze the results, the pair obtained by each scheme will be plotted on a $2D$ point graphs on which the non-dominated set of pairs will be highlighted. In the MOP framework, the strategies within this set can be considered to be the best without defining any order amongst them [69].

V. RESULTS

A. NC PARAMETERIZATION

In this first experimental section, we wish to evaluate how the parameterization of NC affects the accuracy and the efficiency of the conventional *k*NN approach. Although this will also be seen in the next section, the use of efficient techniques for *k*NN does not allow us to properly evaluate these specific issues in detail.

We believe that the most important parameter of the NC configuration is the size of the neural layer from which they are extracted, which represents the dimensionality of the NC itself.

In addition, we have considered the use of the ℓ_2 norm for the normalization of the NC [8]. If x is a vector of size N which represents the NC, the ℓ_2 norm is defined as:

$$|x| = \sqrt{\sum_{k=1}^{N} |x_k|^2}$$

In our preliminary experiments, the use of the ℓ_2 led to a statistically significant improvement in terms of accuracy (Wilcoxon signed-rank test with $\alpha > 0.01$), so we shall from here on report the results regarding this normalization.

Figure 2 shows the results obtained in this first experiment. The evolution of the accuracy with respect to the size of the NC (in logarithmic scale) is depicted. Obviously, when the number of dimensions is very small (*i.e.*, 1, 2, or 4), the NC does not represent the samples at all well and very poor performances are attained. As the dimensionality increases, the growth is very pronounced and quickly stabilizes at around 64 and 128, from which values fluctuate in a less significant manner.

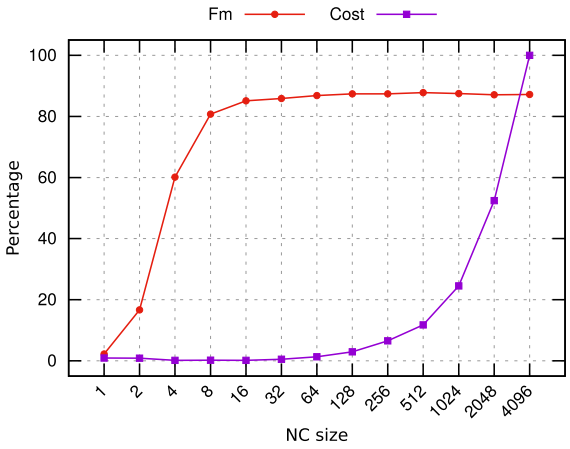


FIGURE 2. Performance of the conventional *k*NN in terms of both accuracy (Fm) and efficiency (cost) with respect to the size of the NC.

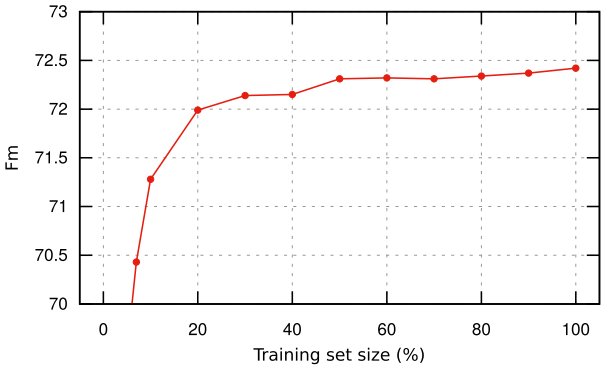


FIGURE 3. Average Fm obtained by varying the size of the training set.

The cost of performing the *k*NN search with respect to the size of the NC is also represented in Fig. 2. As all the results of this experiment have been obtained under the same conditions (same source code and environment), in this case the decision was made to use the time in milliseconds as a measure of cost. Observe, as mentioned previously, that the cost is closely related to the size of the NC since the cost of the distance is proportional to it. The most interesting aspect of this experiment worth noting is that, while increasing the size of NC always increases the computational cost, it is not necessary to use the highest dimensionality to attain the best performance in terms of accuracy.

Another parameter that directly influences both the Fm and the cost of the search is the size of the training set. A trivial way to increase the search efficiency is to reduce the size of this set, however, if the deleted prototypes are not correctly selected by a DR algorithm, this would significantly affect the Fm obtained. As previously argued, the performance of the classifier improves as the training set increases because the likelihood of finding similar prototypes is higher. In addition, it has been demonstrated that its error is bounded by twice the Bayes error when the number of training samples approaches infinity [3]. To evaluate this fact with the datasets considered in this paper, Fig. 3 shows the average Fm obtained by increasing the size of the training sets (note that for this experiment, the prototypes removed from the datasets were selected at random). As can be seen, when the search set is very small (less than 15%) the Fm is abruptly reduced. This result improves as the size increases and, even when the size is close to the total number of samples, a slight upward trend is observed. For this reason, the rest of the experiments will be conducted using the complete training set, with the intention that the improvement attained can only be attributed to the efficient *k*NN search method used.

B. EFFICIENT SEARCH PERFORMANCE

In this section we present the results of the comparison of the different efficient strategies. Since the size of the NC representation might have implications as regards the operation of these strategies, we shall compute the evaluation metrics by taking into account the different NC sizes considered in the previous section. Likewise, for all cases, odd values of k from 1 to 9 will be used for classification. Considering all of the above, the number of experiments comes to a total of 1 600 per dataset. The average values amongst all the dataset-wise results will be reported.

Given this amount of experiments, we shall divide the analysis of the results in such a way that detailed conclusions can be drawn. We shall first carry out a global analysis in which all the strategies of the different families will be evaluated simultaneously, after which we shall present the results by focusing on each individual family.

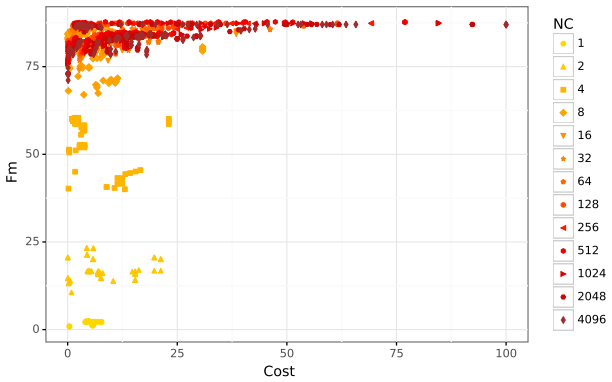


FIGURE 4. Average results of 1, 600 selected experiments in terms of both accuracy (Fm) and efficiency (cost).

1) GENERAL COMPARISON

Figure 4 represents the whole set of experiments carried out. Each strategy for a particular parameter set represents a pair in the (Fm, cost) evaluation space. These pairs are extracted as the averages of the classification experiments with all the datasets, such that the results represent better general trends.

As stated previously, our priority will be the analysis of non-dominated points, since we consider that they represent the optimal set in terms of efficiency and effectiveness within

all the experiments. This is also considered for the sake of the analysis, given that the huge amount of experiments carried out does not allow us to analyze every single result in detail.

The set of non-dominated points is highlighted in the aforementioned figure, in addition to being detailed in Table 4. The family to which each algorithm belongs is also included.

TABLE 4. List of algorithms (with parameters) that belong to the general non-dominated frontier.

Family	Label	NC	Params	Cost	Fm
DR	ECNN	2	k=1; c=1; r=1.0	0.05	20.63
DR	IRB	32	k=7; c=1; r=0.1	0.06	80.49
DR	IRB	128	k=5; c=1; r=0.1	0.07	81.71
DR	1-NE	8	k=5; c=1; r=0.1	0.09	83.17
DR	IRB	8	k=3; c=1; r=0.1	0.12	84.53
ASS	ckNN	128	k=3; c=auto	1.30	86.95
ASS	ckNN	512	k=3; c=auto	1.56	87.35
ASS	ckNN	512	k=7; c=500	2.62	87.54
FSS	KDTree	512	k=3; leaf=10	8.61	87.76

An initial remark to begin with is that the set of non-dominated results is fairly reduced, signifying that few algorithms are really competitive. As expected from the definition of non-dominance, all these results represent different levels of the trade-off between accuracy and efficiency. We can observe a clear trend with respect to the family of algorithms to which they belong. First, we find the ECNN algorithms (NC = 2), IRB (NC = 32, 128), and 1-NE (NC = 8), of the DR algorithm family, which achieve the highest efficiency. In some cases, like the first, this is at the expense of decreasing the Fm to an unacceptable level (20.63). On the opposite side, we find the KDTree algorithm (NC = 512) from the FSS family, which obviously attains the best accuracy by performing an exact search at the cost of having the worst efficiency in the non-dominated set. The ASS family is at the center of both evaluation parameters since it contains algorithms that are more efficient than FSS and more accurate than DR. In this case, a single algorithm ckNN (NC = 128, 512) is found, whose different configurations make it possible to approach either higher effectiveness or higher efficiency.

With regard to the size of NC, we can observe that there is no fully-established regularity. However, the non-dominated results that achieve the highest accuracy generally consider a larger NC, none of which exceeds 512.

2) FAST SIMILARITY SEARCH RESULTS

The results obtained by the algorithms of the FSS family are discussed below. Since these strategies are equal to the conventional kNN in terms of accuracy, the only issue to evaluate here is the efficiency attained. Note that this may lead to confusion given that, by modifying the size of NC and the parameter k, we will obtain different levels of Fm (but all of them have a result that is analogous with all the distances computed). Note that each size of NC considered represents a totally new CNN training.

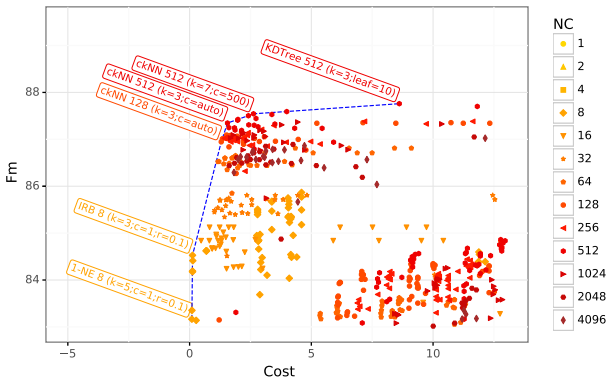


FIGURE 5. Relevant region of the general non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost).

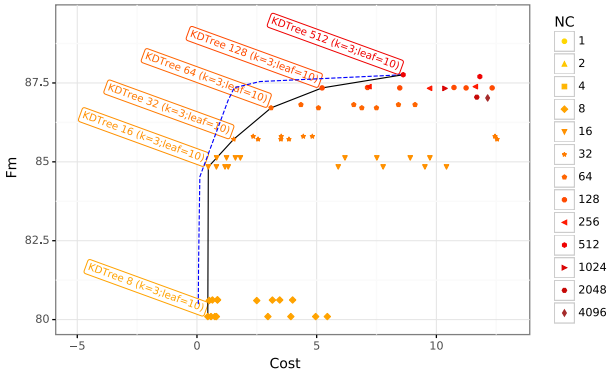


FIGURE 6. Relevant region of the FSS non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost). The blue dashed line represents the general non-dominated frontier.

The results of this single family are shown in Fig. 6, in which the non-dominated points within this set are highlighted. The general non-dominated front is also included as a reference (dashed line) in order to show, in a graphic manner, how the FSS family behaves with respect to the global context. The detailed non-dominated results are presented in Table 5.

TABLE 5. List of FSS algorithms (with parameters) that belong to their specific non-dominated frontier. Results appertaining to the general ND frontier are marked with an asterisk (*).

Label	NC	Params	Cost	Fm
KDTree	8	k=3; leaf=10	0.45	80.10
KDTree	16	k=3; leaf=10	0.47	84.84
KDTree	32	k=3; leaf=10	1.53	85.72
KDTree	64	k=3; leaf=10	3.10	86.71
KDTree	128	k=3; leaf=10	5.23	87.34
KDTree*	512	k=3; leaf=10	8.61	87.76

In the case of the FSS family, all the best results are obtained by the KDTree algorithm, for which different NC values form the non-dominance front. It is interesting to note that, while the computational cost falls proportionally to the

size of NC, the accuracy does not follow such a linear factor. This means that, thanks to the parameterization of the NC, it is possible to attain a relatively low cost without being far from the best accuracy. It should be emphasized, however, that this has a limit because very low values of NC (below 16) lead to a notable loss of accuracy.

In relation to the general results, we can observe that this family only contributes with a single interesting point: that which achieves the best accuracy with a moderate cost. The remaining FSS results are dominated by other configurations.

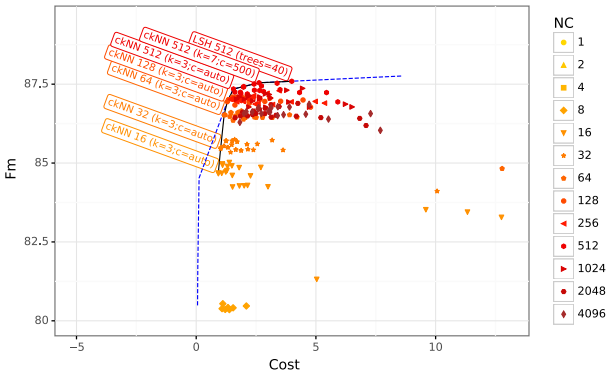


FIGURE 7. Relevant region of the ASS non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost). The blue dashed line represents the general non-dominated frontier.

3) APPROXIMATE SIMILARITY SEARCH RESULTS

As observed in the general comparison, the ASS family is that which has an equable trade-off between accuracy and efficiency. Figure 7 depicts the relevant region of the evaluation space with the non-dominated points of this family of algorithms (detailed in Table 6).

TABLE 6. List of ASS algorithms (with parameters) that belong to their specific non-dominated frontier. Results appertaining to the general ND frontier are marked with an asterisk (*).

Label	NC	Params	Cost	Fm
SH	2	nbits=40	0.19	13.16
ckNN	16	k=3; c=auto	0.92	84.68
ckNN	32	k=3; c=auto	1.02	85.45
ckNN	64	k=3; c=auto	1.17	86.52
ckNN*	128	k=3; c=auto	1.30	86.95
ckNN*	512	k=3; c=auto	1.56	87.35
ckNN*	512	k=7; c=500	2.62	87.54
LSH	512	trees=40	3.99	87.59

Most of the non-dominated set of ASS algorithms originates from the ckNN algorithm, whose parameterization manages to cover almost the entire front. The exceptions are the SH algorithm (NC = 2), which achieves the lowest cost but becomes irrelevant owing to its very poor accuracy, and the LSH algorithm, which slightly improves the accuracy of the best ckNN by a small margin.

Figure 7 also depicts the general non-dominated frontier, in which the goodness of the non-dominated algorithms of this family in relation to the general results will be clearly observed. On the one hand, the rest of the non-dominated front with higher precision barely outperforms them with little margin, and on the other, the non-dominated points with a lower cost begin to show a remarkable drop in terms of Fm. It can, therefore, be concluded that the ASS algorithms have a very interesting trade-off between accuracy and efficiency with respect to the general results.

4) DATA REDUCTION RESULTS

As mentioned above, the DR family appears on the general non-dominance front as a representative of the lowest computational costs. The problem, as stated in Section 2, is that this may lead to a relevant loss of accuracy. Figure 8 depicts a zoomed region of the evaluation space covering the relevant non-dominated front formed by DR algorithms. The details of the results appertaining to that front are provided in Table 7.

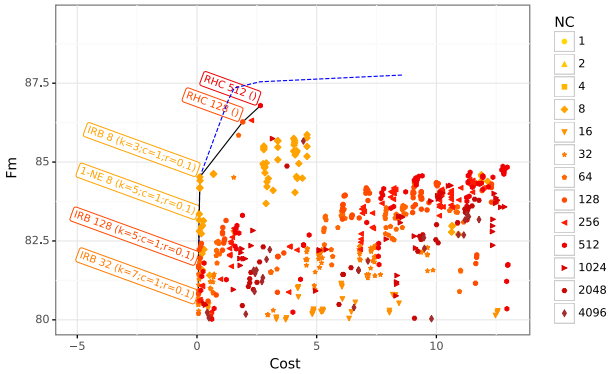


FIGURE 8. Relevant region of the DR non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost). The blue dashed line represents the general non-dominated frontier.

TABLE 7. List of DR algorithms (with parameters) that belong to their specific non-dominated frontier. Results appertaining to the general ND frontier are marked with an asterisk (*).

Label	NC	Params	Cost	Fm
ECNN*	2	k=1; c=1; r=1.0	0.05	20.63
IRB*	32	k=7; c=1; r=0.1	0.06	80.49
IRB*	128	k=5; c=1; r=0.1	0.07	81.71
1-NE*	8	k=5; c=1; r=0.1	0.09	83.17
IRB*	8	k=3; c=1; r=0.1	0.12	84.53
RHC	128		1.91	86.27
RHC	512		2.65	86.79

The figures show a heterogeneous non-dominated front formed by several algorithms. Nevertheless, the differences in their results are rather limited. The front is formed of several results because small increases in cost lead to small increases in accuracy. Given these small differences, the most interesting case may be that of IRB (NC = 8), which obtains the best accuracy (84.53 of Fm) of all the algorithms with a cost below 1%. Note that this algorithm also belongs to the general non-dominated front.

The RHC algorithm is also presented as an interesting alternative within the DR family, as it barely increases the cost and obtains a noticeably higher accuracy. In relation to the general non-dominance front, however, we can observe that this algorithm is dominated by other results with higher accuracy and similar cost (namely ckNN).

C. STATISTICAL FEATURE TRANSFORMATION

In our last series of experiments, we shall evaluate the operation of the efficient algorithms for a kNN search when a statistical transformation of the feature space is applied to the NC. To this end, we consider the Linear Discriminant Analysis (LDA) [70] and the Principal Components Analysis (PCA) [71] algorithms, both of which are usually considered for this purpose [72], [73].

These techniques perform a linear transformation of the data in pursuit of different objectives. LDA seeks a subspace in which the classes in question are better discriminated. PCA, on the other hand, seeks a subspace in which the basis vectors have a maximum variance. One important difference is that LDA is a supervised technique — it requires the labels of each sample — whereas PCA is unsupervised. In our experiments, LDA makes use of the Eigenvalue decomposition with the optimal-shrinkage covariance estimator using Ledoit and Wolf lemma [74], while the PCA implementation automatically selects the number of components such that the amount of variance that needs to be explained is greater than 0.95.

In this case, in order to restrict the number of experiments to be carried out, we have considered only those algorithms that are non-dominated in any of the results reported above. The experiments carried out for all NC sizes of these algorithms were, therefore, repeated by applying LDA and PCA before performing the classification and computing the evaluation measures.

The non-dominated results obtained after applying LDA are shown in Table 8. As occurred in the previous section, the non-dominance front is formed of DR algorithms in the cases of lower cost, of FSS in the cases of higher precision, and of ASS in the intermediate cases. The difference, in this case, is that we can observe results obtained with initially large NC, which are dramatically reduced by the use of LDA.

TABLE 8. List of algorithms (with parameters) that belong to the non-dominated frontier of the LDA mapping.

Family	Label	NC	Params	Cost	Fm	% LDA reduction
DR	RHC	16		0.10	73.50	18.75
DR	RHC	4096		0.12	84.98	99.02
DR	RHC	1024		0.13	85.08	96.08
ASS	ckNN	4096	k=7; c=200	1.14	86.71	99.02
ASS	ckNN	1024	k=7; c=500	1.52	87.08	96.08
FSS	KDTree	1024	k=7; leaf=10	7.14	87.22	96.08

The analogous case with PCA is shown in Table 9. As might be expected of an unsupervised technique, the accuracy is relatively lower than that obtained with LDA. However, the trend is similar: PCA makes it possible to start with larger NCs that are reduced by this technique. It is, however, striking that no FSS representative appears. The most plausible explanation for this is that PCA produces noise, which is better dealt with by non-exact algorithms.

TABLE 9. List of algorithms (with parameters) that belong to the non-dominated frontier of the PCA mapping.

Family	Label	NC	Params	Cost	Fm	% PCA reduction
DR	RHC	32		0.11	76.31	48.13
ASS	ckNN	64	k=7; c=200	1.12	83.98	63.59
ASS	ckNN	32	k=7; c=100	1.15	84.08	48.13
ASS	ckNN	512	k=7; c=500	1.76	85.14	87.97
ASS	LSH	4096	trees=40	2.62	85.67	97.24
DR	ECNN	1024	k=1; c=3; r=1.0	10.72	86.32	92.45

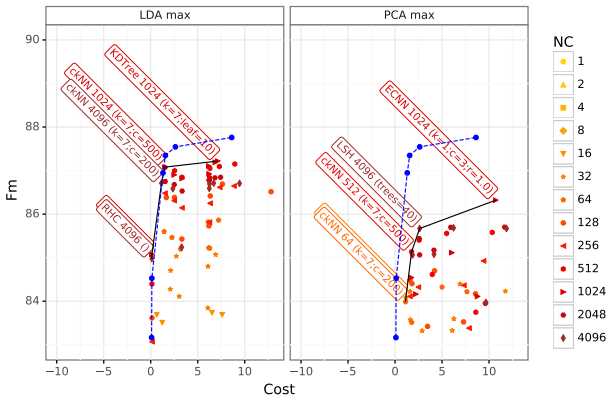


FIGURE 9. Relevant region of the LDA/PCA non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost). The blue dashed line represents the general non-dominated frontier.

About measuring the goodness of applying these transformations with respect to the general results, Fig. 9 shows the area of interest of the results, highlighting both the non-dominated results from LDA and PCA and those originally obtained. Note that neither LDA nor PCA obtain much better results than those originally attained. The case of PCA, whose results are clearly dominated by the general results, is particularly noteworthy. Furthermore, while LDA is competitive, it does not represent a clear improvement in all cases either. Despite not being formally demonstrated, we believe that this may be caused by the use of NC, which already makes a (non-linear) transformation to an appropriate subspace, leaving no room for these techniques to improve the performance. However, if it is necessary to use a pre-trained network that cannot be modified, and whose last hidden layer is large (*eg.* larger than 512 neurons), then it would be appropriate to use this type of techniques, especially LDA.

VI. DISCUSSION

We will jointly analyze the results obtained during experimentation in this section. Based on this analysis, we will provide some "rule of thumb" for the use of specific combinations within the CNN-kNN paradigm according to the needs of the application scenario.

TABLE 10. List of best algorithms (with parameters) that belong to the non-dominated frontier of the NC, LDA and PCA mapping with FSS, ASS and DR families. Table is sorted ascending by Cost value. The global non-dominated are marked with ✓ at the last column.

Type	Family	Algorithm	NC	Params	Cost	Fm	Precision	Recall	Global ND
NC	DR	ECNN	2	k=1; c=1; r=1.0	0.05	20.63	22.04	29.59	✓
NC	DR	IRB	32	k=7; c=1; r=0.1	0.06	80.49	81.32	80.54	✓
NC	DR	IRB	128	k=5; c=1; r=0.1	0.07	81.71	82.57	81.72	✓
NC	DR	1-NE	8	k=5; c=1; r=0.1	0.09	83.17	84.72	83.34	✓
LDA	DR	RHC	16		0.1	73.5	75.59	74.47	
PCA	DR	RHC	32		0.11	76.31	77.03	77.16	
NC	DR	IRB	8	k=3; c=1; r=0.1	0.12	84.53	86.13	84.35	
LDA	DR	RHC	4096		0.12	84.98	85.75	85.11	✓
LDA	DR	RHC	1024		0.13	85.08	85.20	85.57	✓
PCA	ASS	ckNN	64	k=7; c=200	1.12	83.98	84.92	83.75	
LDA	ASS	ckNN	4096	k=7; c=200	1.14	86.71	87.16	86.68	✓
PCA	ASS	ckNN	32	k=7; c=100	1.15	84.08	84.85	83.91	
NC	ASS	ckNN	128	k=3; c=auto	1.3	86.95	87.46	86.83	
LDA	ASS	ckNN	1024	k=7; c=500	1.52	87.08	87.32	87.14	
NC	ASS	ckNN	512	k=3; c=auto	1.56	87.35	87.58	87.41	✓
PCA	ASS	ckNN	512	k=7; c=500	1.76	85.14	85.71	85.10	
NC	ASS	ckNN	512	k=7; c=500	2.62	87.54	87.98	87.58	✓
PCA	ASS	LSH	4096	trees=40	2.62	85.67	85.96	85.61	
LDA	FSS	KDTree	1024	k=7; leaf=10	7.14	87.22	87.44	87.34	
NC	FSS	KDTree	512	k=3; leaf=10	8.61	87.76	88.12	87.76	✓
PCA	DR	ECNN	1024	k=1; c=3; r=1.0	10.72	86.32	86.95	86.23	

To provide an overview, Table 10 shows the set of best algorithms for each representation family (NC, LDA, PCA) from previous sections, where the new non-dominance front has been recalculated (''Global ND'' column). These results are graphically shown in Fig. 10. Note that, in this case, we also add Precision and Recall metrics, in addition to the Cost and Fm. However, it can be observed that, unless few relevant exceptions, both Precision and Recall report very similar figures, and so that the Fm can be reliably used as a summary of the overall accuracy.

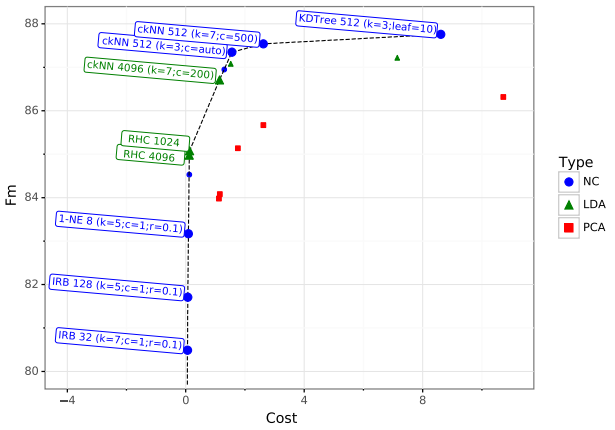


FIGURE 10. Best results of NC, LDA and PCA with non-dominated frontier in terms of both accuracy (Fm) and efficiency (cost). The black dashed line represents the general non-dominated frontier.

The first thing to remark in this final summary is that, among the best configurations (not dominated), we find both representations directly extracted from the neural network (NC) and those obtained after performing the supervised statistical transformation (LDA), while the unsupervised statistical transformation (PCA) is relegated by previous ones.

Furthermore, following the non-dominated front we observe that it navigates through the different families of algorithms, regardless of the representation. The set with the lowest cost is formed by those combinations with DR algorithms, followed by ASS, and then FSS. The representation does have a higher relevance concerning the Fm, as the non-dominated combinations that obtain the best results for this metric use NC.

With the idea of providing a useful analysis for researchers and developers who wish to use the CNN-*k*NN paradigm, we believe that our exhaustive experimentation, objectively summarized in Table 10 and Fig. 10, can provide the following conclusions in this regard:

- When prioritizing efficiency, DR should be used. For example, a good combination is RHC with NC of 1024 dimensions and transformation by LDA (global average: 85.08 of Fm and 0.13 of cost).
- For a trade-off between efficiency and accuracy, the ASS family should be considered. Specifically, our experiments report that ckNN is the most appropriate algorithm, as many of its combinations appear in the global non-dominated front. For example, one might want to use ckNN[k=3;c=auto] with NC of 512 dimensions (global average: 87.35 of Fm and 1.56 of cost).
- If accuracy is the most important criterion, FSS is the right choice because it always reports the highest figures. According to our experiments, KDTree[k=3;leaf=10] with NC of 512 dimensions gets the highest accuracy with reduced cost (global average of 87.76 of Fm and 8.61 of cost).

As seems clear from this list of rules, there is no single algorithm that outperforms the rest in all cases; however, we have been able to draw conclusions that can be generalized to different use-cases, since they have been based on a large

number of experiments, datasets, algorithm combinations, and parameterizations.

A. RUNTIME COST

In our previous sections, the algorithmic cost was used for the evaluation of the efficiency because the runtime is a subjective measure that, as discussed before, depends not only on the underlying hardware but also on the implementation, the programming language, and the libraries used. However, intending to analyze the efficiency improvement more intuitively — using real units (milliseconds) and not percentages with respect to the total cost —, we report in Table 11 a comparison of the runtime. All the experiments were carried out using the Python programming language, and the TensorFlow (v. 1.14) and Scikit-learn (v. 0.20) libraries. The machine used consists of an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz with 23 GB RAM and a Nvidia GeForce GTX 2080 GPU with cuDNN library.

TABLE 11. Search time (in milliseconds) of the *k*NN method (brute force) compared to that of the cKNN method [k=3;c=auto;NC=512] for the different datasets evaluated.

		ms. / query	
Dataset	Training set size	*k*NN	ckNN
USPS	7444	49.24	6.02
MNIST	56004	413.61	2.39
HOMUS	12160	79.41	4.64
NIST	35972	240.69	2.84
CIFAR10	48000	338.94	2.62
CIFAR100	48000	325.47	2.54
MIRBOT100	16275	114.23	4.10
ImageNet	1281167	8687.14	54.72

For this experiment, the ckNN with $k = 3$, $c = auto$ and NC representation of 512 was chosen, since it is one of the global NDs that obtained a more balanced performance in terms of Fm and cost. This table makes a comparison of the results obtained by this algorithm with those obtained by the original kNN method (which performs an exhaustive search), with the intention of comparing the temporal improvement. In addition, the result obtained with a much larger dataset is also included: ImageNet [75], a generic-purpose dataset for object classification used in the Large Scale Visual Recognition Challenge (ILSVRC) with a total of 1,331,167 instances of 224×224 pixels color images divided into 1,000 classes. In this case, the MobileNet v2 [76] network topology was used to extract the NCs. For this, we first initialized the network with the pre-trained weights from the ILSVRC dataset, and then we fine-tuned these weights for the new NC layer size.

As can be seen, the proposed approach is able to reduce the search time significantly in all cases, especially with datasets with a larger search space (i.e., training set size). For example, the search time in MNIST goes from taking almost half a second to just a couple of milliseconds. This difference is even more noticeable if we pay attention to the ImageNet case, since it goes from taking 8.7 seconds to only 54 milliseconds. With datasets with a smaller search space, such as USPS, HOMUS, or MIRBOT100, an improvement is also achieved but not so significant, showing that the efficient search used by ckNN makes the most of large-scale data.

VII. CONCLUSIONS

In this work, we have presented a comprehensive experimental study on the use of an efficient k-Nearest Neighbor search when the space of features is represented by activations of the last layers of a neural network (Neural Codes). The recent advances in neural networks, namely Convolutional Neural Networks (CNN), make this hybrid scheme potentially profitable since higher accuracy can be obtained than with conventional feature extraction processes. This also opens up the possibility of tuning the dimensionality of the feature space, which may affect both the accuracy and the efficiency of the process.

In order to make a comparison that covers most of the possibilities of this hybrid CNN-kNN approach, we have carried out experiments considering several datasets, and many different types of efficient kNN search, which have been grouped into three families of algorithms: Fast Similarity Search (FSS), Approximate Similarity Search (ASS), and Data Reduction (DR).

First, an experiment was conducted in which the impact of the NC size in terms of effectiveness and efficiency on the conventional kNN search was evaluated. It was observed that, although a larger size of NC always proportionally increases the cost, the accuracy does not follow such a linear pattern.

In the case of efficient search algorithms, it was noted that the cost can be greatly reduced with respect to the conventional kNN search. This has been demonstrated in terms of both algorithmic cost and runtime cost — with significant reductions. However, these algorithms often reduce the accuracy of the classification, thereby forming a heterogeneous set of non-dominated results. In general, the non-dominance front is formed of DR algorithms for the lowest costs, of FSS algorithms for the highest accuracies, and of ASS algorithms in the intermediate cases. Concerning the statistical transformations, we have observed that their use does not lead to significant improvements generally; however, the combination of LDA with ASS techniques does produce some optimal combinations.

We believe that this work opens up new perspectives with which to develop efficient search algorithms for kNN. To do this, our goal with respect to future research is to move towards algorithms that are able to perform this search very efficiently without losing accuracy. A good line in this respect would be to include this objective in the CNN loss function, such that the NC not only represent the samples well but also organize themselves better in the NC space.

REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[2] L. I. Kuncheva, "Prototype classifiers and the big fish: The case of prototype (instance) selection," *IEEE Syst., Man, Cybern. Mag.*, vol. 6, no. 2, pp. 49–56, Apr. 2020.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY, USA: Wiley, 2001.

[4] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[5] T. M. Mitchell, *Machine Learning*, New York, NY, USA: McGraw-Hill, 1997.

[6] B. Saçlı, C. Aydınalp, G. Cansız, S. Joof, T. Yilmaz, M. Çayören, B. Önal, and I. Akduman, "Microwave dielectric property based classification of renal calculi: Application of a kNN algorithm," *Comput. Biol. Med.*, vol. 112, Sep. 2019, Art. no. 103366.

[7] F. Jie Huang and Y. LeCun, "Large-scale learning with SVM and convolutional for generic object categorization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2006, pp. 284–291.

[8] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features Off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*. Washington, DC, USA: IEEE Computer Society, Jun. 2014, pp. 512–519.

[9] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, *Neural Codes for Image Retrieval*. Cham, Switzerland: Springer, 2014, pp. 584–599.

[10] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.

[11] P. Jain, B. Kulis, S. Inderjit Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Proc. 21st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, pp. 761–768, 2008.

[12] J. Wang, H. Tao Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," 2014, *arXiv:1408.2927*. [Online]. Available: http://arxiv.org/abs/1408.2927

[13] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Cham, Switzerland: Springer, 2015.

[14] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sep. 1977.

[15] E. V. Ruiz, "An algorithm for finding nearest neighbours in (approximately) constant average time," *Pattern Recognit. Lett.*, vol. 4, no. 3, pp. 145–157, Jul. 1986.

[16] T. Liu, A. W. Moore, and A. Gray, "Efficient exact k-NN and nonparametric classification in high dimensions," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2003, pp. 265–272.

[17] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proc. 23rd Int. Conf. Very Large Data Bases*, 1997, pp. 426–435.

[18] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, J. M. Benítez, and F. Herrera, "Nearest neighbor classification for high-speed big data streams using spark," *IEEE Trans. Syst., Man, Cybern. Syst*, vol. 47, no. 10, pp. 2727–2739, Oct. 2017.

[19] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "KNN-IS: An iterative spark-based design of the k-Nearest neighbors classifier for big data," *Knowl.-Based Syst.*, vol. 117, pp. 3–15, Feb. 2017.

[20] M. Bawa, T. Condie, and P. Ganesan, "LSH forest: Self-tuning indexes for similarity search," in *Proc. 14th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2005, pp. 651–660.

[21] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[22] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[23] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, Jun. 2016.

[24] A.-J. Gallego, J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation," *Pattern Recognit.*, vol. 74, pp. 531–543, Feb. 2018.

[25] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.

[26] L. Nanni and A. Lumini, "Prototype reduction techniques: A comparison among different approaches," *Expert Syst. Appl.*, vol. 38, no. 9, pp. 11820–11828, Sep. 2011.

[27] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 515–516, May 1968.

[28] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 417–435, Mar. 2012.

[29] J. Derrac, C. Cornelis, S. García, and F. Herrera, "Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection," *Inf. Sci.*, vol. 186, no. 1, pp. 73–92, Mar. 2012.

[30] J. Hamidzadeh, R. Monsefi, and H. Sadoghi Yazdi, "IRAHC: Instance reduction algorithm using hyperrectangle clustering," *Pattern Recognit.*, vol. 48, no. 5, pp. 1878–1889, May 2015.

[31] S. Ougiaroglou and G. Evangelidis, "RHC: A non-parametric cluster-based data reduction for efficient *k*-NN classification," *Pattern Anal. Appl.*, vol. 19, no. 1, pp. 93–109, Feb. 2016.

[32] L. Yang, Q. Zhu, J. Huang, and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing*, vol. 230, pp. 427–433, Mar. 2017.

[33] N. García-Pedrajas and A. de Haro-García, "Boosting instance selection algorithms," *Knowl.-Based Syst.*, vol. 67, pp. 342–360, Sep. 2014.

[34] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowl.-Based Syst.*, vol. 39, pp. 240–247, Feb. 2013.

[35] J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Improving kNN multi-label classification in prototype selection scenarios using class proposals," *Pattern Recognit.*, vol. 48, no. 5, pp. 1608–1622, May 2015.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[37] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic ConvNet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, Sep. 2016.

[38] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems (NIPS)*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Cambridge, MA, USA: MIT Press, 2014, pp. 3320–3328.

[39] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *CoRR*, vol. abs/1405.3531, Aug. 2014.

[40] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2017.

[41] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Heidelberg, Germany: Springer, 2010, pp. 177–186.

[42] J. Walters-Williams and Y. Li, "Comparative study of distance functions for nearest neighbors," in *Advanced Techniques in Computing Sciences and Software Engineering*, K. Elleithy, Ed. Dordrecht, The Netherlands: Springer, 2010, pp. 79–84.

[43] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 538–543.

[44] S.-H. Yoon, S.-W. Kim, and S. Park, "C-rank: A link-based similarity measure for scientific literature databases," *Inf. Sci.*, vol. 326, pp. 25–40, Jan. 2016.

[45] M. Zhang, J. Wang, and W. Wang, "HeteRank: A general similarity measure in heterogeneous information networks by integrating multi-type relationships," *Inf. Sci.*, vol. 453, pp. 389–407, Jul. 2018.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1–9.

[47] I. Wald and V. Havran, "On building fast kd-trees for ray tracing, and on doing that in O(N log N)," in *Proc. IEEE Symp. Interact. Ray Tracing*, Sep. 2006, pp. 61–69.

[48] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. London, U.K.: Chapman & Hall, 2017.

[49] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.

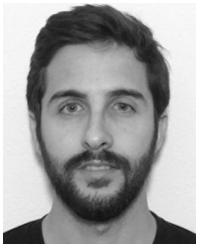
[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[51] J. Calvo-Zaragoza and J. Oncina, "Recognition of pen-based music notation: The HOMUS dataset," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Aug. 2014, pp. 3038–3043.

[52] R.-A. Wilkinson, J. C. Geist, S. A. Janet, P. Grother, C. J. C. Burges, R. H. Creecy, B. Hammond, J. J. Hull, N. W. Larsen, T. P. Vogl, and C. L. Wilson, *The First Census Optical Character Recognition System Conference*. Washington, DC, USA: US Department of Commerce, 1992.

[53] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, USA, Tech. Rep., 2009.
[54] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.
[55] A. Pertusa, A.-J. Gallego, and M. Bernabeu, "MirBot: A collaborative object recognition system for smartphones using convolutional neural networks," *Neurocomput.*, vol. 293, pp. 87–99, Jun. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218302704, doi: 10.1016/j.neucom.2018.03.005.
[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, Apr. 2014.
[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[58] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 1800–1807.
[59] J. Calvo-Zaragoza, A.-J. Gallego, and A. Pertusa, "Recognition of handwritten music symbols with convolutional neural codes," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 691–696.
[60] A.-J. Gallego, A. Pertusa, and J. Calvo-Zaragoza, "Improving convolutional neural Networks' accuracy in noisy environments using k-nearest neighbors," *Appl. Sci.*, vol. 8, no. 11, p. 2086, Oct. 2018.
[61] N. Papernot and D. Patrick McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *CoRR*, vol. abs/1803.04765, Mar. 2018.
[62] B. V. Dasarathy, J. S. Sánchez, and S. Townsend, "Nearest neighbour editing and condensing tools–synergy exploitation," *Pattern Anal. Appl.*, vol. 3, no. 1, pp. 19–30, Feb. 2000.
[63] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1450–1464, Nov. 2007.
[64] J. R. Rico-Juan and J. M. Iñesta, "New rank methods for reducing the size of the training set using the nearest neighbor rule," *Pattern Recognit. Lett.*, vol. 33, no. 5, pp. 654–660, Apr. 2012.
[65] P. Hernandez-Leal, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. A. Olvera-Lopez, "InstanceRank based on borders for instance selection," *Pattern Recognit.*, vol. 46, no. 1, pp. 365–375, Jan. 2013.
[66] D. R. Wilson and R. T. Martinez, "Instance pruning techniques," in *Proc. 14th Int. Conf. Mach. Learn. (ICML)*. San Francisco, CA, USA: Morgan Kaufmann, 1997, pp. 403–411.
[67] H. Brighton and C. Mellish, "On the consistency of information filters for lazy learning algorithms," in *Principles of Data Mining and Knowledge Discovery* (Lecture Notes in Computer Science), vol. 1704, J. M. Żytkow and J. Rauch, Eds. Berlin, Germany: Springer, 1999, pp. 283–288.
[68] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, Nov. 2013.
[69] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer, 1999.
[70] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, vol. 544. Hoboken, NJ, USA: Wiley, 2004.
[71] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
[72] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego, CA, USA: Academic, 1990.
[73] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, 2nd Quart., 2001.
[74] O. Ledoit and M. Wolf, "Honey, I shrunk the sample covariance matrix," *J. Portfolio Manage.*, vol. 30, no. 4, pp. 110–119, Jul. 2004.
[75] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[76] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

ANTONIO-JAVIER GALLEGO received the B.Sc. and M.Sc. degrees in computer science and the Ph.D. degree in computer science and artificial intelligence from the University of Alicante, Spain, in 2004 and 2012. He is currently an Assistant Professor with the Department of Software and Computing Systems, University of Alicante. He has been a Researcher on ten research projects funded by both the Spanish Government and private companies. He has authored or coauthored 45 works published in international journals, conferences, books, and book chapters. His research interests include deep learning, pattern recognition, and computer vision.



JORGE CALVO-ZARAGOZA received the Ph.D. degree in computer science from the University of Alicante, Spain, in June 2016. He then joined McGill University, Canada, and Universitat Politecnica de Valencia, Spain, as a Postdoctoral Fellow, in 2017 and 2018, respectively. Since 2019, he has been an Assistant Professor and a Researcher with the University of Alicante. He has authored more than 50 articles in peer-reviewed journals and international conferences. His main interests are focused on pattern recognition and computer vision applied to document analysis.

JUAN RAMÓN RICO-JUAN received the Ph.D. degree from the University of Alicante, Spain, in 2001. He is currently a Lecturer with the University of Alicante and a Researcher with the Pattern Recognition and Artificial Intelligence Group. His research interests include areas such as pattern recognition and machine learning, as well as working on structured data learning, distance editing, prototype selection and generation, and deep neural networks. He has participated in eight national projects which have resulted in the publication in 21 high impact journals (JCR) and 17 in international conferences.